

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Applicant: Julian S. Taylor	Patent No.: 7,107,267 B2
App. No.: 10/066,362	Issue Date: September 12, 2006
Filed: January 31, 2002	Con. No.: 2482
Title: METHOD, SYSTEM, PROGRAM AND DATA STRUCTURE FOR IMPLEMENTING A LOCKING MECHANISM FOR A SHARED RESOURCE	Art Unit: 2165
	Examiner: Hicks, Michael J.

**REQUEST FOR CERTIFICATE OF CORRECTION  
UNDER 37 C.F.R. § 1.322**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

Applicant hereby requests a Certificate of Correction under 37 C.F.R. § 1.322 be issued for the above patent, in accordance with the attached request.

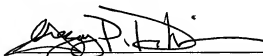
The Patent Office has made several typographical errors in the claims of the issued patent. A copy of the allowed claims is attached for review.

All errors sought to be corrected were made by the Patent Office. Therefore, the Assignee believes no fee is due with this filing. However, should any fees or petitions be required, please consider this a request therefor and authorization to charge Deposit Account No. 04-1415 as necessary.

Should the Examiner have any questions, please contact the undersigned attorney.

Dated: 24 OCT 2006

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'Gregory P. Durbin', is written over a horizontal line.

Gregory P. Durbin, Registration No. 42,503  
Attorney for Applicant  
USPTO Customer No. 20686

DORSEY & WHITNEY LLP  
Republic Plaza Building, Suite 4700  
370 Seventeenth Street  
Denver, Colorado 80202-5647  
Phone: (303) 629-3400  
Fax: (303) 629-3450

**UNITED STATES PATENT AND TRADEMARK OFFICE**  
**CERTIFICATE OF CORRECTION**

**PATENT NO.** : **7,107,267 B2**  
**ISSUE DATE** : **September 12, 2006**  
**INVENTOR(S)** : **Julian S. Taylor**

**It is certified that an error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:**

In column 16, at line 35, insert --in-- after "file".

In column 16, at line 42, delete "far" and insert --for--.

In column 18, at line 12, insert --the-- between "to" and "first".

In column 18, at line 20, after "updating", delete "to" and insert --the--.

In column 22, at line 45, delete "alter", and insert --after--.

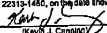
Mailing address of Sender:

Dorsey & Whitney LLP  
Republic Plaza Building, Suite 4700  
370 Seventeenth Street  
Denver, CO 80202-5647  
Phone: (303) 629-3400  
Fax: (303) 629-3450  
USPTO Customer No.: 20696

Patent No. 7,107,267 B2

FEB 23 2006

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being transmitted by facsimile to the Patent and Trademark Office, facsimile no. (571) 273-8300 at MS Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below.

Dated: February 23, 2006 Signature:   
(Kevin J. Canning)

Docket No.: SMQ-119/P6157  
(PATENT)**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Patent Application of:  
Julian S. Taylor

Application No.: 10/066362

Confirmation No.: 2482

Filed: January 31, 2002

Art Unit: 2165

For: METHOD, SYSTEM, AND PROGRAM AND  
DATA STRUCTURE FOR IMPLEMENTING A  
LOCKING MECHANISM FOR A SHARED  
RESOURCE

Examiner: M. J. Hicks

**AMENDMENT IN RESPONSE TO NON-FINAL OFFICE ACTION**

MS Amendment  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Dear Sir:

**INTRODUCTORY COMMENTS**

In response to the Office Action dated December 27, 2005, please amend the above-identified U.S. patent application as follows:

**Amendments to the Claims** are reflected in the listing of claims which begins on page 2 of this paper.

**Remarks/Arguments** begin on page 19 of this paper.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

**AMENDMENTS TO THE CLAIMS**

1. (Currently Amended) A method for implementing a locking mechanism to control access to a shared resource, comprising:

receiving a request to access the shared resource;  
determining whether a first file has a first name;  
renaming the first file to a second name if the first file has the first name;  
updating a second file to indicate the received request in a queue of requests to the shared resource if the first file is renamed to the second name, wherein an ordering of the requests in the queue is used to determine whether access to the shared resource is granted to the request;  
reading and buffering a checksum from the second file before updating the second file;  
and

renaming the first file to the first name after the second file is updated.

2. (Currently Amended) The method of claim 1, wherein updating the second file further comprises:

~~reading and buffering a checksum from the second file before updating the second file;~~  
copying the second file to a temporary file;  
updating the temporary file to indicate the received request; and  
if the buffered checksum and checksum in the second file match, then replacing the second file with the updated temporary file.

3. (Original) The method of claim 2, further comprising:

calculating a revised checksum from the updated temporary file; and  
including the revised checksum in the updated temporary file before replacing the second file with the updated temporary file.

4. (Original) The method of claim 2, further comprising:

returning retry if the buffered checksum and checksum in the second file do not match,  
wherein the second file is not replaced with the updated temporary file if there is no match.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

5. (Currently Amended) The method of claim 1, wherein the check sum is maintained in both the first file and the second file and the updating the second file further comprises:  
reading and buffering a checksum calculated from the second file before updating the second file, wherein the checksum is maintained in both the first file and the second file;  
copying the second file to a temporary file, wherein the temporary file is updated to indicate the received request; and  
if the buffered checksum and the checksum in both the first file and second file all match, then replacing the second file with the updated temporary file.
6. (Original) The method of claim 5, further comprising:  
calculating a revised checksum from the updated temporary file;  
including the revised checksum in the updated temporary file before replacing the second file with the temporary file; and  
including the revised checksum in the first file before renaming the first file to the first name.
7. (Original) The method of claim 1, wherein a lease data structure indicates at least one request indicated in the queue in the second file granted access to the shared resource and, for each request granted access to the shared resource, a lease time during which the request is granted access to the shared resource, further comprising:  
determining whether one request in the queue is permitted access to the shared resource based on the ordering of the requests in the queue;  
updating the lease data structure to indicate the determined request and the lease time if the determined request is permitted access to the shared resource; and  
returning a message to the determined request indicating that access to the shared resource is granted and the lease time during which access is granted.
8. (Original) The method of claim 7, wherein the request to access is for exclusive access to the shared resource, wherein determining whether one request in the queue is permitted access to the shared resource further comprises:  
determining the request following the request at a top of the queue after determining that the lease time has expired;

Application No.: 10/066362

Docket No.: SMQ-119/P6157

removing the request at the top of the queue;  
updating the queue to indicate the determined request at the top of the queue; and  
updating the lease data structure to identify the determined request and set a new lease time for the determined request in the lease data structure during which the request has exclusive access to the shared resource.

9. (Original) The method of claim 8, wherein the queue and lease structure are updated after renaming the first file from the first name to the second name.

10. (Original) The method of claim 7, wherein the lease data structure comprises a file and wherein updating the lease data structure further comprises:

reading and buffering a checksum from the lease data structure;  
copying the lease data structure to a temporary lease file, wherein the temporary lease file is updated to indicate the determined request and the new lease time; and  
if the buffered checksum and checksum in the lease data structure match, then replacing the lease data structure with the temporary lease file.

11. (Original) The method of claim 7, wherein the request to access is for non-exclusive access to the shared resource and wherein the lease data structure indicates a number of requests allowed simultaneous access to the shared resource, wherein determining whether one considered request in the queue is permitted access to the shared resource further comprises:

determining whether a number of current readers is less than the allowed readers;  
determining whether less than the number of current readers precedes the considered request in the queue;

updating the lease data structure to identify the considered request and set a new lease time for the considered request to have non-exclusive access to the shared resource if the number of current readers is less than the allowed readers and less than the number of current readers precedes the considered request in the queue; and

incrementing the number of current readers after updating the lease data structure to indicate the considered request.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

12. (Original) The method of claim 11, further comprising:

- removing one request for non-exclusive access from the queue whose lease time has expired;
- removing indication from the lease data structure of the request removed from the queue;
- and
- decrementing a field in the lease data structure indicating the number of current readers.

13. (Original) The method of claim 11, wherein the queue is capable of including entries for exclusive and non-exclusive access to the shared resource, and wherein the lease data structure is updated to indicate the considered request and set a new least time for the considered request to have non-exclusive access to the shared resource if there is no exclusive access request between a top of the queue and the considered request.

14. (Original) The method of claim 7, further comprising:

- returning a retry message to the request indicating to retry the request to access the shared resource and the lease time if the request is determined not to be permitted access to the shared resource, wherein the request is retried after the lease time has expired.

15. (Original) The method of claim 1, further comprising:

- receiving a request to remove an entry from the queue;
- determining whether the first file has the first name;
- renaming the first file to the second name if the first file has the first name; updating the second file to indicate that the request is removed from the queue; and
- renaming the first file to the first name after the second file is updated.

16. (Original) The method of claim 1, wherein the locking mechanism is capable of being executed on multiple operating system platforms, and wherein the steps of renaming and updating correspond to native operating system commands implemented across operating system platforms.



Application No.: 10/066362

Docket No.: SMQ-119/P6157

17. (Original) The method of claim 16, wherein the locking mechanism is implemented in a cross-platform computer programming language that is called by applications seeking to access the shared resource.

18. (Original) The method of claim 17, wherein the cross-platform computer programming language comprises Java.

19. (Original) The method of claim 1, wherein the shared resource comprises a data structure and wherein the access comprises one of read or write access to the shared data structure.

20. (Currently Amended) A system for implementing a locking mechanism to control access to a shared resource, comprising:

a file system;

a processor in communication with ~~the a~~ first computer readable medium and file system; code implemented in the first computer readable medium executed by the processor to

cause the processor to perform:

(i) receiving a request to access the shared resource;

(ii) determining whether a first file in the file system has a first name;

(iii) renaming the first file to a second name if the first file has the first name;

(iv) updating a second file in the file system to indicate the received request in a queue of requests to the shared resource if the first file is renamed to the second name, wherein an ordering of the requests in the queue is used to determine whether access to the shared resource is granted to the request;

(v) reading and buffering a checksum from the second file in the computer readable medium before updating the second file; and

(vi) renaming the first file to the first name after the second file is updated.

21. (Currently Amended) The system of claim 20, further comprising:

a computer readable medium,

wherein the code for updating the second file further causes the processor to perform:

Application No.: 10/066362

Docket No.: SMQ-119/P6157

~~(i) reading and buffering a checksum from the second file in the computer-readable medium before updating the second file;~~  
(ii) copying the second file to a temporary file;  
(iii) updating the temporary file to indicate the received request; and  
(iiiw) if the buffered checksum and checksum in the second file match, then replacing the second file with the updated temporary file.

22. (Original) The system of claim 20, further comprising:

a computer readable medium;  
a lease data structure in the computer readable medium indicating at least one request indicated in the queue in the second file granted access to the shared resource and, for each request granted access to the shared resource, a lease time during which the request is granted access to the shared resource;

wherein the code further causes the processor to perform:

(i) determining whether one request in the queue is permitted access to the shared resource based on the ordering of the requests in the queue;  
(ii) updating the lease data structure to indicate the determined request and the lease time if the determined request is permitted access to the shared resource; and  
(iii) returning a message to the determined request indicating that access to the shared resource is granted and the lease time during which access is granted.

23. (Original) The system of claim 22, wherein the request to access is for exclusive access to the shared resource, wherein the code causing the processor to determine whether one request in the queue is permitted access to the shared resource further causes the processor to perform:

determining the request following the request at a top of the queue after determining that the lease time has expired;

removing the request at the top of the queue;

updating the queue to indicate the determined request at the top of the queue; and

updating the lease data structure to identify the determined request and set a new lease time for the determined request in the lease data structure during which the request has exclusive access to the shared resource.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

24. (Original) The system of claim 22, wherein the request to access is for non-exclusive access to the shared resource and wherein the lease data structure indicates a number of requests allowed simultaneous access to the shared resource, wherein the code causing the processor to determine whether one considered request in the queue is permitted access to the shared resource further causes the processor to perform:

determining whether a number of current readers is less than the allowed readers;

determining whether less than the number of current readers precedes the considered request in the queue;

updating the lease data structure to identify the considered request and set a new lease time for the considered request to have non-exclusive access to the shared resource if the number of current readers is less than the allowed readers and less than the number of current readers precedes the considered request in the queue; and

incrementing the number of current readers after updating the lease data structure to indicate the considered request.

25. (Original) The system of claim 20, wherein the code further causes the processor to perform:

receiving a request to remove an entry from the queue; determining whether the first file has the first name;

renaming the first file to the second name if the first file has the first name; updating the second file to indicate that the request is removed from the queue; and

renaming the first file to the first name after the second file is updated.

26. (Original) The system of claim 20, wherein the locking mechanism is capable of being executed on multiple operating system platforms, and wherein the steps of renaming and updating correspond to native operating system commands implemented across operating system platforms.

27. (Original) The system of claim 20, wherein the shared resource comprises a data structure and wherein the access comprises one of read or write access to the shared data structure.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

28. (Currently Amended) A system for implementing a locking mechanism to control access to a shared resource, comprising:

- a file system implemented including a first file and a second file;
- means for receiving a request to access the shared resource;
- means for determining whether the first file has a first name;
- means for renaming the first file to a second name if the first file has the first name;
- means for updating the second file to indicate the received request in a queue of the

requests to the shared resource if the first file is renamed to the second name, wherein an ordering of the requests in the queue is used to determine whether access to the shared resource is granted to the request and wherein the means for updating the second file further performs reading and buffering a checksum from the second file in the computer readable medium before updating the second file; and

means renaming the first file to the first name after the second file is updated.

29. (Currently Amended) The system of claim 28, further comprising:

- a computer readable medium;
- wherein the means for updating the second file further performs:

(i) reading and buffering a checksum from the second file in the computer readable medium before updating the second file;

(ii) copying the second file to a temporary file in the file system;

(iii) updating the temporary file to indicate the received request; and

(i)iv) if the buffered checksum and checksum in the second file match, then replacing the second file with the updated temporary file.

30. (Currently Amended) The system of claim 28 wherein the checksum is maintained in both the first file and the second file and the system; further comprising:

- a computer readable medium;
- wherein the means for updating the second file further performs:

(i) reading and buffering a checksum in the computer readable medium calculated from the second file before updating the second file, wherein the checksum is maintained in both the first file and the second file;

Application No.: 10/066362

Docket No.: SMQ-119/P6157

- (ii) copying the second file to a temporary file in the file system, wherein the temporary file is updated to indicate the received request; and
- (iv) if the buffered checksum and the checksum in both the first file and second file all match, then replacing the second file with the updated temporary file.

31. (Original) The system of claim 28, further comprising:

- a computer readable medium;
- a lease data structure in the compute readable medium that indicates at least one request indicated in the queue in the second file granted access to the shared resource and, for each request granted access to the shared resource, a lease time during which the request is granted access to the shared resource;
- means for determining whether one request in the queue is permitted access to the shared resource based on the ordering of the requests in the queue;
- means for updating the lease data structure to indicate the determined request and the lease time if the determined request is permitted access to the shared resource; and
- means for returning a message to the determined request indicating that access to the shared resource is granted and the lease time during which access is granted.

32. (Original) The system of claim 31, wherein the request to access is for exclusive access to the shared resource, wherein the means for determining whether one request in the queue is permitted access to the shared resource further performs:

- determining the request following the request at a top of the queue after determining that the lease time has expired;
- removing the request at the top of the queue;
- updating the queue to indicate the determined request at the top of the queue; and
- updating the lease data structure to identify the determined request and set a new lease time for the determined request in the lease data structure during which the request has exclusive access to the shared resource.

33. (Original) The system of claim 31, wherein the lease data structure comprises a file and wherein the means for updating the lease data structure further performs:

- reading and buffering a checksum from the lease data structure;

Application No.: 10/066362

Docket No.: SMQ-119/P6157

copying the lease data structure to a temporary lease file, wherein the temporary lease file is updated to indicate the determined request and the new lease time; and

if the buffered checksum and checksum in the lease data structure match, then replacing the lease data structure with the temporary lease file.

34. (Original) The system of claim 31, wherein the request to access is for non-exclusive access to the shared resource and wherein the lease data structure indicates a number of requests allowed simultaneous access to the shared resource, wherein the means for determining whether one considered request in the queue is permitted access to the shared resource further performs:

determining whether a number of current readers is less than the allowed readers;

determining whether less than the number of current readers precedes the considered request in the queue;

updating the lease data structure to identify the considered request and set a new lease time for the considered request to have non-exclusive access to the shared resource if the number of current readers is less than the allowed readers and less than the number of current readers precedes the considered request in the queue; and

incrementing the number of current readers after updating the lease data structure to indicate the considered request.

35. (Original) The system of claim 31, further comprising:

means for returning a retry message to the request indicating to retry the request to access the shared resource and the lease time if the request is determined not to be permitted access to the shared resource, wherein the request is retried after the lease time has expired.

36. (Original) The system of claim 28, further comprising:

means for receiving a request to remove an entry from the queue;

means for determining whether the first file has the first name;

means for renaming the first file to the second name if the first file has the first name;

means for updating the second file to indicate that the request is removed from the queue;

and

means for renaming the first file to the first name after the second file is updated.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

37. (Original) The system of claim 28, wherein the locking mechanism is capable of being executed on multiple operating system platforms, and wherein the means for renaming and updating utilize native operating system commands implemented across operating system platforms.

38. (Original) The system of claim 37, wherein the locking mechanism is implemented in a cross-platform computer programming language that is called by applications seeking to access the shared resource.

39. (Original) The system of claim 28, wherein the shared resource comprises a data structure and wherein the access comprises one of read or write access to the shared data structure.

40. (Currently Amended) An article of manufacture including code for implementing a locking mechanism to control access to a shared resource, wherein the code causes operations to be performed comprising:

- receiving a request to access the shared resource;
- determining whether a first file has a first name;
- renaming the first file to a second name if the first file has the first name;
- updating a second file to indicate the received request in a queue of requests to the shared resource if the first file is renamed to the second name, wherein an ordering of the requests in the queue is used to determine whether access to the shared resource is granted to the request;
- reading and buffering a checksum from the second file before updating the second file;
- and
- renaming the first file to the first name after the second file is updated.

41. (Currently Amended) The article of manufacture of claim 40, wherein updating the second file further comprises:

- reading and buffering a checksum from the second file before updating the second file;
- copying the second file to a temporary file;
- updating the temporary file to indicate the received request; and

Application No.: 10/066362

Docket No.: SMQ-119/P6157

if the buffered checksum and checksum in the second file match, then replacing the second file with the updated temporary file.

42. (Original) The article of manufacture of claim 41, further comprising:

calculating a revised checksum from the updated temporary file; and

including the revised checksum in the updated temporary file before replacing the second file with the updated temporary file.

43. (Original) The article of manufacture of claim 41, further comprising:

returning retry if the buffered checksum and checksum in the second file do not match, wherein the second file is not replaced with the updated temporary file if there is no match.

44. (Currently Amended) The article of manufacture of claim 40, wherein the checksum is maintained in both the first file and the second file and the updating the second file further comprises:

reading and buffering a checksum calculated from the second file before updating the second file, wherein the checksum is maintained in both the first file and the second file;

copying the second file to a temporary file, wherein the temporary file is updated to indicate the received request; and

if the buffered checksum and the checksum in both the first file and second file all match, then replacing the second file with the updated temporary file.

45. (Original) The article of manufacture of claim 44, further comprising:

calculating a revised checksum from the updated temporary file;

including the revised checksum in the updated temporary file before replacing the second file with the temporary file; and

including the revised checksum in the first file before renaming the first file to the first name.

46. (Original) The article of manufacture of claim 40, wherein a lease data structure indicates at least one request indicated in the queue in the second file granted access to



Application No.: 10/066362

Docket No.: SMQ-119/P6157

the shared resource and, for each request granted access to the shared resource, a lease time during which the request is granted access to the shared resource, further comprising:

- determining whether one request in the queue is permitted access to the shared resource based on the ordering of the requests in the queue;

- updating the lease data structure to indicate the determined request and the lease time if the determined request is permitted access to the shared resource; and

- returning a message to the determined request indicating that access to the shared resource is granted and the lease time during which access is granted.

47. (Original) The article of manufacture of claim 46, wherein the request to access is for exclusive access to the shared resource, wherein determining whether one request in the queue is permitted access to the shared resource further comprises:

- determining the request following the request at a top of the queue after determining that the lease time has expired;

- removing the request at the top of the queue;

- updating the queue to indicate the determined request at the top of the queue; and

- updating the lease data structure to identify the determined request and set a new lease time for the determined request in the lease data structure during which the request has exclusive access to the shared resource.

48. (Original) The article of manufacture of claim 47, wherein the queue and lease structure are updated after renaming the first file from the first name to the second name.

49. (Original) The article of manufacture of claim 46, wherein the lease data structure comprises a file and wherein updating the lease data structure further comprises:

- reading and buffering a checksum from the lease data structure;

- copying the lease data structure to a temporary lease file, wherein the temporary lease file is updated to indicate the determined request and the new lease time; and

- if the buffered checksum and checksum in the lease data structure match, then replacing the lease data structure with the temporary lease file.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

50. (Currently Amended) The article of manufacture of claim ~~46~~<sup>4656</sup>, wherein the request to access is for non-exclusive access to the shared resource and wherein the lease data structure indicates a number of requests allowed simultaneous access to the shared resource, wherein determining whether one considered request in the queue is permitted access to the shared resource further comprises:

- determining whether a number of current readers is less than the allowed readers;
- determining whether less than the number of current readers precedes the considered request in the queue;
- updating the lease data structure to identify the considered request and set a new lease time for the considered request to have non-exclusive access to the shared resource if the number of current readers is less than the allowed readers and less than the number of current readers precedes the considered request in the queue; and
- incrementing the number of current readers after updating the lease data structure to indicate the considered request.

51. (Original) The article of manufacture of claim 50, further comprising:

- removing one request for non-exclusive access from the queue whose lease time has expired;
- removing indication from the lease data structure of the request removed from the queue;
- and
- decrementing a field in the lease data structure indicating the number of current readers.

52. (Original) The article of manufacture of claim 50, wherein the queue is capable of including entries for exclusive and non-exclusive access to the shared resource, and wherein the lease data structure is updated to indicate the considered request and set a new least time for the considered request to have non-exclusive access to the shared resource if there is no exclusive access request between a top of the queue and the considered request.

53. (Original) The article of manufacture of claim 46, further comprising:

- returning a retry message to the request indicating to retry the request to access

Application No.: 10/066362

Docket No.: SMQ-119/P6157

the shared resource and the lease time if the request is determined not to be permitted access to the shared resource, wherein the request is retried after the lease time has expired.

54. (Original) The article of manufacture of claim 40, further comprising:

receiving a request to remove an entry from the queue;

determining whether the first file has the first name;

renaming the first file to the second name if the first file has the first name; updating the second file to indicate that the request is removed from the queue; and

renaming the first file to the first name after the second file is updated.

55. (Original) The article of manufacture of claim 40, wherein the locking mechanism is capable of being executed on multiple operating system platforms, and wherein the steps of renaming and updating correspond to native operating system commands implemented across operating system platforms.

56. (Original) The article of manufacture of claim 55, wherein the locking mechanism is implemented in a cross-platform computer programming language that is called by applications seeking to access the shared resource.

57. (Original) The article of manufacture of claim 56, wherein the cross-platform computer programming language comprises Java.

58. (Original) The article of manufacture of claim 40, wherein the shared resource comprises a data structure and wherein the access comprises one of read or write access to the shared data structure.

59. (Currently Amended) A computer readable medium including data structures for implementing a locking mechanism to control access to a shared resource, comprising:  
a first file having one of a first name or a second name, wherein in response to receiving a request to access the shared resource, the first file is renamed to the second name if the first file has the first name; and

Application No.: 10/066362

Docket No.: SMQ-119/P6157

a second file indicating a queue of requests to the shared resource, wherein an ordering of the requests in the queue is used to determine whether access to the shared resource is granted to the request, and wherein the second file is updated to indicate the received request in the queue when the first file is renamed to the second name, wherein the first file is renamed to the first name after the second file is updated; and

a checksum field in the second file, wherein the checksum is buffered before updating the second file.

60. (Currently Amended) The computer readable medium of claim 59, further comprising:

~~a checksum field in the second file, wherein the checksum is buffered before updating the second file; and~~

a temporary file, wherein the second file is copied to the temporary file and the temporary file is updated to indicate the received request, wherein the second file is replaced with the updated temporary file if the buffered checksum and checksum in the second file match.

61. (Original) The computer readable medium of claim 59, further comprising:

a lease data structure indicating at least one request indicated in the queue in the second file granted access to the shared resource and, for each request granted access to the shared resource, a lease time during which the request is granted access to the shared resource, wherein determining whether one request in the queue is permitted access to the shared resource is based on the ordering of the requests in the queue, and wherein the lease data structure is updated to indicate the determined request and the lease time if the determined request is permitted access to the shared resource.

62. (Original) The computer readable medium of claim 61, wherein the lease data structure comprises a file, further comprising:

a checksum field within the lease data structure, wherein the checksum from the checksum field is buffered when updating the lease data structure; and

a temporary lease file, wherein updating the lease data structure involves copying the lease data structure to the temporary lease file, wherein the temporary lease file is updated to indicate the determined request following the request at the top of the queue and the new lease

Application No.: 10/066362

Docket No.: SMQ-119/P6157

time, wherein if the buffered checksum and checksum in the lease data structure match, then the lease data structure is replaced with the temporary lease file.

63. (Original) The computer readable medium of claim 61, wherein the request to access is for non-exclusive access to the shared resource, further comprising:

a number of allowed readers allowed simultaneous access to the shared resource indicated in the lease data structure, wherein the lease data structure is updated to identify the considered request and set a new lease time for the considered request to have nonexclusive access to the shared resource if a number of current readers is less than the allowed readers and less than the number of current readers preceding the considered request in the queue, wherein the number of current readers is incremented after updating the lease data structure to indicate the considered request.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

**REMARKS**

Applicant amends claims 1, 2, 5, 20, 21, 28, 29, 30, 40, 41, 44, 50, 59, and 60. No new matter is added. Upon entry of this amendment, claims 1-63 are pending, of which claims 1, 20, 28, 40, and 59 are independent. Applicant notes with appreciation that the Examiner deems claims 2-6, 10, 21, 29-30, 33, 41-45, 29, 60, and 62 to recite patentable subject matter. Applicant respectfully submits that the pending claims define over the art of record.

**Claim Objection**

Claim 50 is objected to due to minor informalities. Applicant amends claim 50 to address the Examiner's concern. Applicant respectfully requests that the Examiner reconsider and withdraw the claim objection.

**Claim Rejection Under 35 U.S.C. §112**

Claim 20 is rejected under 35 U.S.C. §112 second paragraph as being indefinite. Applicant amends claim 20 to address the Examiner's concern. Applicant respectfully requests that the Examiner reconsider and withdraw the rejection to claim 20.

**Claim Rejection Under 35 U.S.C. §103****Claims 1, 7-9, 14, 16-17, 19-20, 22-23, 26-28, 31-32, 35, 37-40, 46-48, 53, 55-56, 58-59, and 61**

Claims 1, 7-9, 14, 16-17, 19-20, 22-23, 26-28, 31-32, 35, 37-40, 46-48, 53, 55-56, 58-59, and 61 are rejected under 35 U.S.C. §103(a) as being unpatentable over United States Patent No. 6,145,006 to Vishlitzky et al. (hereafter "Vishlitzky") in view of "Improving Performance by Use of Adaptive Objects: Experimentation with a Configurable Multiprocessor Thread Package" by Schwan et al. (hereafter "Schwan"). Applicant respectfully submits that the combination of the Vishlitzky reference and the Schwan reference fail to teach or suggest the limitation of reading and buffering a checksum from the second file before updating the second file, as required by amended independent claims 1, 20, 28, 40, and 59.

The Vishlitzky reference teaches the use of a resource manager within the storage system to manage the sharing of common resources. However, the Vishlitzky reference does not teach

Application No.: 10/066362

Docket No.: SMQ-119/P6157

or suggest the limitation of reading and buffering a checksum from a file before updating the file, as required by the amended independent claims 1, 20, 28, 40, and 59. The Schwan reference fails to bridge the factual deficiency of the Vishlitzky reference.

The Schwan reference teaches how to improve the performance of parallel programs by customizing operating system functionalities. However, the Schwan reference also fails to teach or suggest the limitation of reading and buffering a checksum from a file before updating the file, as required by the amended independent claims 1, 20, 28, 40, and 59.

Additionally, there is no motivation to combine the Vishlitzky reference with the Schwan reference. The Vishlitzky reference focuses on how to provide a universal lock mechanism so that regardless of the type of host computers, one host computer is able to lock out other host computers from accessing certain common resources. See Col. 1, lines 60-67. The Vishlitzky reference achieves this objective by providing a resource manager within the storage system and not on the host computers. See Col. 2, lines 7-10. In contrast, the Schwan reference considers the customization of operating system functionality important to improve performance for specific application programs. See Abstract. Hence for multiple host computers with different operating systems, it becomes cumbersome to write separate customization functionalities for different operating systems to access common resources. Therefore, there is no motivation for one of ordinary skill in the art to modify the Vishlitzky reference with the teachings of the Schwan reference.

Accordingly, Applicant respectfully submits that the combination of the Vishlitzky reference and the Schwan reference do not teach or suggest the limitation of the limitation of reading and buffering a checksum from a file before updating the file, as required by the amended independent claims 1, 20, 28, 40, and 59. Applicant respectfully requests that the Examiner reconsider and withdraw the rejection of independent claims 1, 20, 28, 40, and 59 and their corresponding dependent claims 7-9, 14, 16-17, 19, 22-23, 26-27, 31-32, 35, 37-39, 46-48, 53, 55-56, 58, and 61.

Claims 11-13, 24, 34, 50-52, and 63

Claims 11-13, 24, 34, 50-52, and 63 are rejected under 35 U.S.C. §103(a) as being

Application No.: 10/066362

Docket No.: SMQ-119/P6157

unpatentable over Vishlitzky in view of Schwan in further view of "A semi-Optimistic Database Scheduler Based on Commit Ordering" by Taterinov et al. (hereafter "Taterinov") and "Class readwrite lock: public abstractsemaphore" (hereafter "DS"). Applicant respectfully submits that the combination of the Vishlitzky reference, the Schwan reference, the Taterinov reference, and the DS reference do not teach or suggest the limitation of reading and buffering a checksum from a file before updating the file, as required by the amended independent claims 1, 20, 28, 40, and 59, which claims 11-13, 24, 34, 50-52, and 63 depend.

As set forth above, the Vishlitzky reference and the Schwan reference do not teach or suggest the limitation of reading and buffering a checksum from a file before updating the file. Applicant respectfully submits that the Taterinov reference and the DS reference fail to bridge this factual deficiency.

The Taterinov reference teaches a database scheduler that does not block a write on a data item when another transaction holds a read lock on the same data item. However, the Taterinov reference does not teach or suggest the limitation of reading and buffering a checksum from a file before updating the file, as required by claims 11-13, 24, 34, 50-52, and 63.

The DS reference also fails to teach or suggest the limitation of reading and buffering a checksum from a file before updating the file. The DS reference teaches a class for a read/write lock and its corresponding methods. However, the DS reference does not teach or suggest reading and buffering a checksum as required by the claimed invention.

Accordingly, the combination of the Vishlitzky reference, the Schwan reference, the Taterinov reference, and the DS reference do not teach or suggest the limitation of reading and buffering a checksum as required by claims 11-13, 24, 34, 50-52, and 63. Applicant respectfully requests that the Examiner reconsider and withdraw the rejection of claims 11-13, 24, 34, 50-52, and 63.

Claims 15, 25, 36, and 54

Claims 15, 25, 36, and 54 are rejected under 35 U.S.C. §103(a) as being unpatentable over Vishlitzky in view of "A Fair, Fast Scalable Reader-Writer Lock" by Krieger et al.



Application No.: 10/066362

Docket No.: SMQ-119/P6157

(hereafter "Krieger"). Applicant respectfully submits that the combination of the Vishlitzky reference and the Krieger reference do not teach or suggest the limitation of reading and buffering a checksum from a file before updating the file, as required by amended independent claims 1, 20, 28, and 40, which claims 15, 25, 36, and 54 depend.

As set forth above, the Vishlitzky reference does not teach or suggest the limitation of reading and buffering a checksum from a file before updating the file. Applicant respectfully submits that the Krieger reference also fails to teach or suggest this limitation.

The Krieger reference teaches a scalable reader-writer lock by using a doubly linked list for readers and when a reader is releasing the lock, the reader synchronize with its nearest neighbors to remove itself from the list. However, the Krieger reference does not teach or suggest reading and buffering a checksum as required by the claimed invention.

Accordingly, Applicant respectfully submits that the combination of the Vishlitzky reference and the Krieger reference do not teach or suggest reading and buffering a checksum from a file before updating the file, as required by claims 15, 25, 36, and 54. Applicant respectfully requests that the Examiner reconsider and withdraw the rejection of claims 15, 25, 36, and 54.

#### Claims 18 and 57

Claims 18 and 57 are rejected under 35 U.S.C. §103(a) as being unpatentable over Vishlitzky in view of Schwan and further in view of "Design issues for efficient implementation of MPI in Java" by Judd et al. (hereafter "Judd"). Applicant respectfully submits that the combination of the Vishlitzky reference, the Schwan reference, and the Judd reference do not teach or suggest the limitation of reading and buffering a checksum from a file before updating the file, as required by independent claims 1 and 54, which claims 18 and 57 depend.

As set forth above, the combination of the Vishlitzky reference and the Schwan reference fail to teach or suggest the limitation of reading and buffering a checksum from a file before updating the file. Applicant respectfully submits that the Judd reference fails to bridge this factual deficiency.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

The Judd reference teaches an implementation of MPI using the Java programming language. However, the Judd reference does not teach or suggest the limitation of reading and buffering a checksum as required by the claimed invention. Additionally, as set forth above, there is no motivation to combine the Vishlitzky reference and the Schwan reference. Applicant further submits that there is also no motivation to modify the teachings of the Schwan reference with the Judd reference as the Schwan reference focuses on customization on different operating systems to improve performance, whereas the Judd reference teaches a cross-platform computer programming language that disregard the performance issue on different platforms.

Accordingly, Applicant respectfully submits that the combination of the Vishlitzky reference, the Schwan reference, and the Judd reference do not teach or suggest the limitation of reading and buffering a checksum as required by claims 18 and 57. Applicant respectfully requests that the Examiner reconsider and withdraw the rejection of claims 18 and 57.

Application No.: 10/066362

Docket No.: SMQ-119/P6157

CONCLUSION

In view of the above amendment, Applicant believes the pending application is in condition for allowance.

Applicant believes no fee is due with this statement. However, if a fee is due, please charge our Deposit Account No. 12-0080, under Order No. SMQ-119 from which the undersigned is authorized to draw.

Dated: February 23, 2006

Respectfully submitted,

By 

Kevin J. Canning

Registration No.: 35,470

LAHIVE &amp; COCKFIELD, LLP

28 State Street

Boston, Massachusetts 02109

(617) 227-7400

(617) 742-4214 (Fax)

Attorney For Applicant